201年10月13日 星期二

Compte Rendu IA01 TD04

1. Définir formellement le problème :

- Ensemble d'états : i ∈ Ens := {1,..., 20}∪{Entree,Sortie}
- Etats initiaux : {Entree}
- Etats solutions : {Sortie}
- Ensemble d'actions : déplacements autorisés

2. Représentation du labyrinthe

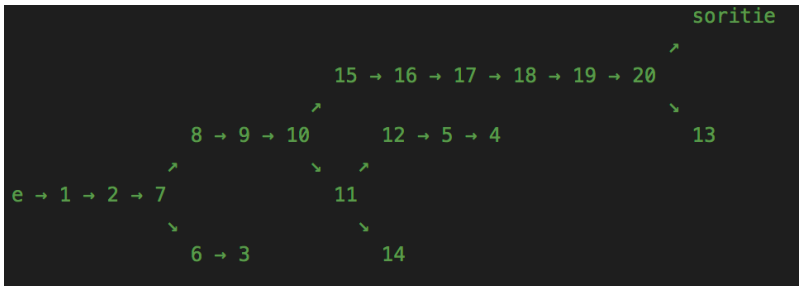
```
(setq *lab* '(
  (Entree 1)
  (1 Entree 2)
  (2 1 7)
  (3 6)
  (4 5)
  (5 4 12)
  (6 3 7)
  (7 2 6 8)
  (8 7 9)
  (9 8 10)
  (10 9 11 15)
  (11 10 12 14)
  (12 5 11)
  (13 20)
  (14 11)
  (15 10 16)
  (16 15 17)
  (17 16 18)
  (18 17 19)
  (19 18 20)
  (20 13 19 Sortie)
  (Sortie 20)
  )
)
```

On choisit la représentation sous forme de clé-valeurs :

((clé1 valeur11 valeur 12 ... valeur 1m) ... (clén valeurn1 ...valeur nm))

Avec ici la clé qui représente un noeud à chaque fois un état. Les valeurs associés sont les états accessible depuis l'état clé. En outre n = card(Ens) = 22, m ∈ {1,2,3,4} car m dépend de la clé, c'est à dire selon la clé on peut bien savoir son successeur.

3. Définition l'arbre de recherche:



4. Effectuer manuellement les parcours en profondeur d'abord er largeur d'abord.

En profondeur d'abord:

Entree-1-2-7-6-3-8-9-10-11-14-12-5-4-15-16-17-18-19-20-13-sortiee

En largeur d'abord:

Entree-1-2-7-6-8-3-9-10-11-15-14-12-16-5-17-4-18-19-20-13-sortiee

5. Fonction explore:

```
(assoc 'c '( (a 1) ( b 2) (c 3 4)))
>(c 3 4)
```

Fonction pour chercher le successeur d'un clé:

```
(defun successeurs (etat lab)
    (cdr(assoc etat lab))
)
```

Il faut aussi avoir une liste qui peut stocker les clés déjà visités:`(setq *visited* '())` après visiter un clé, lui push dans la liste visited.
Une fois on a un e liste visited, quand on parcourt d'un noeud on peut tester s'il déjà visité ou pas.

```
(defun successeurs_valides (etat lab deja_visites)
    (let ((successeurs (successeurs etat lab)) (retour '()))
        (dolist (node successeurs)
            (if (not (member node deja_visites))
            (push node retour)
            )
        )
        (reverse retour)
    )
)
```

```
(defun explore_dfs (etat lab chemin)
    (pushnew etat *visitesd* )
    (if (eq etat 'sortie)
        (print chemin)
        (let ((successeurs (successeurs_valides etat lab
*visitesd* ) ))
            (dolist (node successeurs)
                (explore_dfs node lab (append chemin
(list node)) )
            )
        )
    )
)
```

Résultat:

```
YanLIUdeMacBook-Pro:~ yann$ clisp ./Lisp/TD04.lisp

((ENTREE 1) (1 ENTREE 2) (2 1 7) (3 6) (4 5) (5 4 12) (6 3 7) (7 2 6 8) (8 7 9)
 (9 8 10) (10 9 11 15) (11 10 12 14) (12 5 11 14) (13 20) (14 11) (15 10 16)
 (16 15 17) (17 16 18) (18 17 19) (19 18 20) (20 19 SORTIE) (SORTIE 20))
(1 2 7 8 9 10 15 16 17 18 19 20 SORTIE)
```